UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Desarrollo de un sistema de medida de recogida de Datos y monitorización de tráfico VoIP

Alberto Martín Hernández

Tutor: Jorge E. López de Vergara Méndez

JULIO 2014

Desarrollo de un sistema de medida de recogida de Datos y monitorización de tráfico VoIP

AUTOR: Alberto Martín Hernández

TUTOR: Jorge E. López de Vergara Méndez

High Performance Computing and Networking Research Group

Dpto. de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2014

***Resumen* —** Debido al enorme crecimiento y evolución que están sufriendo las redes de telecomunicación, se hace necesario un seguimiento y monitorización del uso que se hace de dichas redes. Este crecimiento se ha hecho notar en el tráfico que circula sobre el protocolo IP, como es el cado de la Voz sobre IP (VoIP) y la Televisión sobre IP (IPTV). Los servicios de este tipo se están masificando y cada vez va a más, debido a la mejora del rendimiento de las redes IP. Esto hace que el control y seguimiento del tráfico tengan que perfeccionarse para buscar una solución óptima. Para resolver este problema tienen que desarrollarse disectores de los protocolos monitorizados, que deben de ser capaces de obtener la información necesaria para poder llevar a cabo una buena monitorización. Así es como surge el proyecto VoIPCallMon, cuyo objetivo es poder monitorizar diversos protocolos de VoIP en redes de alta velocidad con sondas de bajo coste. Pudiendo así, disponer de un gran número de sondas a un precio bajo para poder obtener información de la red en más puntos de rastreo. Dentro de este proyecto, existen módulos especializados en diseccionar un determinado protocolo. En este documento se explicará cómo se ha conseguido dar solución a la disección de un protocolo de señalización de VoIP que pertenece a la empresa Avaya, UNISTIM. Este protocolo es utilizado, en la mayoría de casos, en redes corporativas, donde es verdaderamente necesario obtener información sobre las llamadas realizadas. La solución que se propone, no solo rastrea información sobre la señalización, sino que también recopila información sobre el flujo RTP, dando la posibilidad de grabar las conversaciones que se produjeron. Todo esto, siempre, adaptando el módulo creado al proyecto que se encuentra por encima. En resumen, esta solución muestra cómo realizar una disección eficiente a nivel de software, además de entregar la información necesaria para llevar a cabo una buena monitorización.

***Palabras clave*—** Voz sobre IP, UNISTIM, disección, monitorización, IP, protocolo propietario, VoIPCallMon.

***Abstract* —** The sustained high growth and evolution of telecommunication networks has led to the need of monitoring its use. This growth is noticeable in the IP traffic, such as Voice over IP (VoIP) or IP Television (IPTV). This type of services is being massive, and they are growing because of the improvements of the IP networks performance. Then, the traffic control and monitoring has to be improved to find an optimal solution. To solve this issue, it is necessary to develop dissectors for the monitored protocols, which have to be able to obtain the needed information to carry out a good monitoring. This is how the VoIPCallMon project appears. Its objective is to monitor several VoIP protocols in high speed networks with low-cost off-the-shelf probes. In this way, it is possible to have a large number of low-cost probes to obtain network information in several vantage points. In this project there are specialized modules to dissect a concrete protocol. In this document we will explain how we could solve the dissection of UNISTIM, a VoIP signaling protocol owned by Avaya. This protocol is used, mostly, in enterprise networks, where it is truly necessary to obtain information about the calls made or received. The proposed solution traces information about the signaling, and also gathers information about the RTP flow, being able to record the phone conversations. All this has to be adapted as a module for the VoIPCallMon project. In summary, this development shows how to perform an efficient protocol dissection in software, as well as providing the needed information to get a good monitoring.

***Index terms* —** Voice over IP, UNISTIM, dissection, monitoring, IP, proprietary protocol, VoIPCallMon.

Agradecimientos

Una vez redactado este documento y vividas todas las experiencias en su realización es cuando se puede echar la vista atrás. Es muy complicado agradecer y mencionar a todas las personas que me han ayudado y han hecho posible que yo esté aquí.

En primer lugar, a las personas que me han dado todo lo que tenían para conseguir que llegase aquí, mi familia. Sin duda, tanto mi padre, como mi madre, como mi hermano, son ejemplos a seguir en mi día a día y de los nunca ha faltado apoyo, en todos los aspectos, y sé que seguiré teniéndolo. Sin duda, todas las metas que consiga superar es por ellos.

Al tutor de este TFG, además de ser mi tutor durante toda la carrera, Jorge. Sin duda, sin él no habría sido posible, debido a su generosidad ayudándome siempre que le requería y buscando como mejorar, no solo durante el TFG sino durante toda la carrera. También a David por su paciencia conmigo en el desarrollo del TFG, resolviéndome dudas conceptuales y de programación.

A mis compañeros de la universidad, que más que compañeros son amigos, que han hecho mucho más amena la carrera. Nacho, Sergio, Tito, Sergio, Eduardo, Alfredo, Andrea y Roi. Especial agradecimiento a Bárbara, ya que ha tenido que ayudarme, animarme en los malos momentos, ser la mejor compañía y soportar mis peores momentos, haciéndome pasar a su vez los mejores momentos de esta etapa.

Por último, dedicar el último párrafo a mi grupo de amigos que han hecho posible una total desconexión de todo lo relacionado con la universidad en momentos cruciales, y que llevan conmigo mucho más tiempo que lo que ha durado esta etapa. A Juan Carlos, Miguelillo, Daniel y Javier, y por supuesto, no puedo, ni podré, olvidarme de lo que fue la persona que más tiempo ha convivido conmigo como amigo y con la que más vivencias he tenido. Digo la que fue, porque durante este viaje de cuatro años nos has dejado, pero esto también es tuyo, también te pertenece. Gracias a ti también Jota.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

GLOSARIO

ACK *Acknowledgement* (asentimiento), confirmación de un mensaje que fue enviado desde el destino hacia el origen.

CMD Comando.

HPCN *High Performance Computing and Networking researching group* (grupo de investigación de Redes y Computación de Altas Prestaciones).

HTTP *Hypertext Transfer Protocol* (Protocolo de Transferencia de Hipertexto)**,** protocolo usado en las transacciones en Internet.

IETF *Internet Engineering Task Force* (grupo de trabajo de Ingeniería de Internet), entidad reguladora de los estándares en Internet.

IP *Internet Protocol* (Protocolo de Interredes), protocolo de comunicación clasificado en la capa de red según el modelo OSI.

ITU *International Telecommunication Union*, Unión Internacional de Telecomunicaciones

MAC *Media Access Control* (Control de Acceso al Medio), dirección que corresponde de manera única a una tarjeta o dispositivo de red

PBX *Private Branch eXchange***,** centralita telefónica

PSTN *Public Switched Telephone,* Network, Red Telefónica Pública Conmutada

QoE *Quality of Experience,* calidad de la experiencia

QoS *Quality of Service,* calidad del servicio

RTCP *RTP Control Protocol* (Protocolo de Control de RTP), Protocolo que proporciona información de control asociado al flujo RTP

RTP *Real- time Transport Protocol* (Protocolo de Transporte de Tiempo Real), protocolo utilizado para la transmisión de información en tiempo real

TCP *Transmission Control Protocol* (Protocolo de Control de la Transmisión), protocolo de transporte que garantiza que los datos serán entregados

UDP *User Datagram Protocol* (Protocolo de Datagramas de Usuario), protocolo del nivel de transporte basado en el intercambio de datagramas

VLAN *Virtual LAN* (Red de Área Local Virtual), método para crear redes lógicas independientes dentro de una misma red física.

VoIP *Voice over IP* (Voz sobre IP), grupo de recursos que posibilita que viaje la señal de voz a través de Internet, utilizando IP.

1 Introducción

1.1 Introducción

A medida que avanzan los años, el uso de las redes IP se ha ido incrementando. La mejora en rendimiento, velocidad y comprensión de los datos hace que las empresas en telefonía y telecomunicaciones centren sus esfuerzos en su utilización. Éste aumento de popularidad es debido a que permiten transmitir todo tipo de información a tiempo real de una manera cada vez más eficaz y más rápidamente.

Dicha mejora hace que la voz transmitida sobre IP (VoIP, *Voice over IP*) sea, y vaya a ser, toda una revolución en el mundo de las telecomunicaciones. Debido a su enorme potencial, la VoIP hará que cambiemos nuestra idea de telefonía tal y como la hemos conocido durante los últimos cien años.

La tecnología de VoIP, hasta hace relativamente poco, ha sido percibida como poco más que un método para la obtención y realización de llamadas a larga distancia más baratas. Pero la realidad es que permite a la voz llegar a ser una aplicación más dentro de la red de datos, permitiendo a la gente la posibilidad de comunicarse de una manera más flexible y sin limitaciones.

Un indicador claro del aumento del uso de la transmisión de voz sobre redes IP es, que cada vez existen más protocolos de VoIP. Incluso empresas privadas crean sus propios protocolos según sus necesidades, encontrando así en el mercado actual un gran número de protocolos de VoIP, tanto propietarios como abiertos al público. Dichos protocolos, se utilizan para la señalización entre terminales y el Gestor de Llamadas o *Call Manager*. Así se consigue mantener información sobre las llamadas que se realizan. Éste hecho hace que sea fundamental poder monitorizar dicha comunicación para poder hacer el seguimiento de la comunicación y obtener información, como direcciones de origen y destino, puertos utilizados, códecs de audio y estado de la llamada.

Dos ejemplos prácticos muy claros de aplicación de la Voz sobre IP son *Korea Telecom*, que tenía 2,1 millones de subscritores en el año 2011 [1], y en Italia, unos de los principales proveedores de VoIP, disponía de más de 600 000 en 2010 [2]. Esto hace que sea necesaria una monitorización exhaustiva de los enlaces para poder dar así, una calidad parecida a la red de telefonía actual.

El problema que existe para monitorizar todos éstos protocolos es que son distintos entre ellos y hace que sea necesario establecer un disector diferente para cada uno. Cada protocolo tiene sus propias características y peculiaridades, y cada uno de ellos mantiene un tipo de información según el uso que se le quiera dar al propio enlace. Dichas peculiaridades pueden ir desde el tipo de transporte que utilizan, es decir, TCP o UDP hasta el más mínimo detalle en la señalización.

Centrándonos ya en el caso de este Trabajo Fin de Grado (TFG), el protocolo UNISTIM, podemos decir que es un protocolo de señalización desarrollado por la empresa Nortel y que fue adquirido por Avaya posteriormente. Es un protocolo propietario utilizado por varias empresas para telefonía corporativa.

La principal motivación de este TFG es, por tanto, conseguir monitorizar y diseccionar el protocolo de voz sobre IP creado por Nortel, UNISTIM. De esta manera, será posible obtener información sobre las comunicaciones de VoIP que se hayan realizado en un determinado punto del enlace.

1.2 Objetivos del Trabajo Fin de Grado

El principal objetivo de éste TFG es la creación y desarrollo de un disector que sea capaz de obtener ciertas medidas y datos de una conexión entre dos puntos del protocolo de VoIP UNISTIM. Se medirá el tiempo de duración de la llamada, tanto en el origen como en el destino, además de extraer la información se mencionó en el apartado anterior.

Todo surge de un proyecto de desarrollo del grupo de investigación HPCN de la Escuela Politécnica Superior, en la Universidad Autónoma de Madrid, llamado VoIPCallMon. Éste proyecto tiene como fin poder monitorizar y diseccionar diversos tipos de conexiones y enlaces según el protocolo de señalización de VoIP que se use, de una manera eficiente para poder ser utilizable en redes de altas prestaciones.

Por ello, el contexto de este trabajo es más amplio, no es tan solo crear y desarrollar dicho disector, sino también hacer que sea una parte del proyecto VoIPCallMon. Es decir, tiene que conseguir ser una utilidad más dentro de un proyecto mucho mayor, teniendo así que adaptarse a sus prestaciones y estructuras.

Por lo tanto, la idea principal de éste TFG evoluciona hasta llegar a crear una aplicación capaz de obtener información completa de un enlace UNISTIM, pero además sabiendo que debe formar parte de un proyecto mayor que busca eficiencia, rapidez de procesado y bajo coste, para poder usarlo en dispositivos de propósito general.

Por otro lado, al ser un protocolo de señalización de Voz sobre IP también sería casi obligado obtener información sobre el flujo RTP. Con eso alcanzaríamos una monitorización de las llamadas más completa.

En suma, los principales objetivos de este TFG son:

- Crear una aplicación lo más eficiente posible que sea capaz de obtener la información necesaria del tráfico de VoIP para poder conocer el estado de las conexiones desde el punto en el que queramos diseccionar para el protocolo UNISTIM.

- Conseguir adaptar dicha aplicación al proyecto existente VoIPCallMon, teniendo en cuenta sus estructuras y sus capacidades. Consiguiendo así, que sea un código eficaz, fácil de procesar y de mantener.

- Conseguir monitorizar los flujos RTP, tanto información como contenido, a través de los datos obtenidos en el protocolo de señalización UNISTIM.

1.3 Fases de realización

Para el desarrollo de éste proyecto ha sido necesario seguir una serie de fases para intentar llegar al mejor resultado posible:

- **Documentación:** La primera fase fue documentarse e investigar sobre el estado del arte y sobre el propio protocolo UNISTIM, para poder ver, y conocer, todos sus entresijos y poder así, realizar el modelo a posteriori. Dentro de ésta fase también se hizo necesario investigar sobre cómo crear un entorno de pruebas e implementar uno después de la investigación. Este paso fue necesario para poder obtener información sobre el protocolo a través de pruebas con capturas reales existentes.

- **Análisis:** Reunida toda la información necesaria, se pasó a realizar un análisis sobre el comportamiento del protocolo. Con el fin de adquirir los conocimientos necesarios para realizar unas buenas disección y monitorización y crear así un modelo para nuestra aplicación.

- **Desarrollo:** Una vez conocido y estudiado el protocolo entramos en la segunda fase, la implementación de la aplicación. Éste fue sin duda el grueso de la realización del TFG ya que tiene dos partes bien diferenciadas. Por un lado, estaba el estudio del proyecto VoIPCallMon para poder adaptar el objetivo del TFG a dicho proyecto, y por otro lado, se encuentra el propio desarrollo de la aplicación.

- **Validación:** Una vez desarrollada la aplicación, entramos en la tercera fase. Ésta última fase consistía en probar la aplicación desarrollada. Para ello, tuvimos que crear nuestro propio banco de pruebas, pero con unas dimensiones mayores al banco de la primera fase. Una vez tuviéramos el banco montado, haríamos diversas pruebas con varias conexiones, para así comprobar que nuestra aplicación se comporta de la forma esperada.

- **Redacción de la memoria:** Por último, se redactó el documento que se muestra para poder plasmar el conocimiento y la experiencia adquirida en este TFG.

1.4 Estructura del documento

En este apartado se muestra cómo va a estar estructurado este Trabajo Fin de Grado:

1. En el siguiente capítulo se realizará una exposición del Estado del arte, que ofrecerá una visión de otros protocolos de señalización sobre VoIP, algunos incluidos en VoIPCallMon, y así poderlos relacionar con el tratado de este documento. También haremos un tratamiento especial al protocolo de UNISTIM, para que se pueda comprender como funciona. Además se introducirá el funcionamiento del proyecto VoIPCallMon de manera resumida. Con el fin de poder comprender cómo surgió y para que surgió.

2. Después de explicar cómo funciona, enumeraremos y explicaremos en el capítulo 3, el funcionamiento y los requisitos que tiene que cumplir nuestra aplicación. Los requisitos serán tanto los funcionales como los no funcionales. Es decir, se realizará un análisis más técnico de la solución que se realiza en este documento.

3. Una vez visto que es lo que se quiere conseguir, en el capítulo 4, se explicará cómo hemos desarrollado el módulo de UNISTIM. Haciendo especial hincapié a la lógica implementada para monitorizar el flujo. Explicando también, que parte de la funcionalidad de VoIPCallMon es la que realiza UNISTIM y cómo se integra en el proyecto.

4. En el capítulo 5 se pondrá la aplicación a prueba. Se hará en varios escenarios de llamadas.

5. Por último, en el capítulo 6 se verán las conclusiones del trabajo realizado, así como las aportaciones futuras que se pueden añadir al trabajo presentado.

2 Estado del arte

2.1 Introducción

Este capítulo comenzará introduciendo de manera resumida el funcionamiento y los elementos básicos de un escenario de Voz sobre IP. Posteriormente se introducirán otros tipos de protocolos de señalización de VoIP y se presentará el estado del arte actual. Así se podrá tener una visión más global del estado de las comunicaciones que funcionan bajo voz sobre IP. Además se detallará el funcionamiento del protocolo a diseccionar, UNISTIM. Para finalizar el capítulo, se expondrá de manera resumida el funcionamiento del proyecto VoIPCallMon. Esto nos ayudará a poder ubicar nuestro módulo a crear.

2.2 La Voz sobre IP

La Voz sobre protocolo de Internet, también llamado Voz sobre IP (VoIP), es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP. Esto significa que se envía la señal de voz en forma digital, en paquetes de datos, en lugar de enviarla en forma analógica a través de circuitos utilizables sólo por telefonía convencional como las redes PSTN (sigla de *Public Switched Telephone Network*, Red Telefónica Pública Conmutada) [3]. Los elementos que dispone una red de VoIP son los siguientes:

- **El cliente:** El cliente establece y origina las llamadas realizadas de voz. Esta información se recibe a través del micrófono del usuario y se codifica, se empaqueta y, de la misma forma, esta información se decodifica y reproduce a través de los altavoces o audífonos.
- **El servidor (PBX):** Se encarga de manejar operaciones de base de datos. Entre estas operaciones se tienen la contabilidad, la recolección, el enrutamiento, la administración y control del servicio, el registro de los usuarios, etc.
- ***Gateways*:** brinda un puente de comunicación entre todos los usuarios. Su función principal es la de proveer interfaces con la telefonía tradicional adecuada. Es la parte que hace posible que esa llamada que viene por Internet logre conectarse con un cliente de una empresa telefónica fija o móvil.

Vistos los elementos, se pasará a ver la estructura típica utilizada en la VoIP. Para que se vea mejor nos basaremos en la siguiente figura:

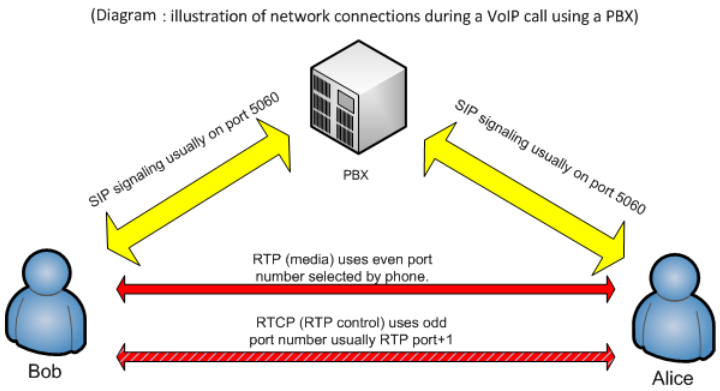

Figura I: Diagrama de flujo típico VoIP [4]

En la Figura I, se puede ver el diagrama de los flujos de señalización (amarillo) y RTP (rojo) más usual. En este caso, el protocolo de señalización es SIP. El protocolo de señalización es utilizado para establecer la conexión entre ambos puntos. Para ello, se tienen que apoyar en un Gestor de Llamadas, que es el encargado de rutar la llamada. Una vez se establecen los parámetros de configuración del flujo de audio, éste comienza a fluir. En la Figura I es el flujo rojo, el RTP. Los puertos del RTP suelen ser configurados por los clientes, aunque también hay casos en los que los configura el Gestor. Por otro lado, se encuentra el flujo que aparece en rojo, pero con líneas transversales. Es el flujo RTCP, encargado de mantener información de control sobre el flujo de audio.

Como podemos ver, el flujo de audio circula de extremo a extremo, mientras que la señalización es dirigida por el Gestor. Este es el caso más habitual, pero como veremos adelante, existe el caso en el que el flujo RTP también transcurre por el Gestor.

Una vez se tienen las ideas básicas sobre el funcionamiento y los elementos básicos de la Voz sobre IP, se pasará a ver distintos tipos de señalización.

2.3 Protocolos de señalización de VoIP

La señalización en VoIP tiene un papel fundamental en la red, ya que es la encargada de establecer, mantener, administrar y finalizar una conversación entre dos puntos. También es la que se encarga de supervisar la llamada y de proveer QoS en cada canal de transmisión. A continuación se describirá algunos de los protocolos más importantes utilizados en VoIP [5]:

- H.323: es una familia de estándares desarrollado por la ITU en 1996 con el objetivo de ofrecer un mecanismo de transporte para servicios multimedia sobre redes que no garantizan QoS. Al principio fue definido y creado como un protocolo de videoconferencia pero ha ido evolucionando rápidamente para cubrir todos los requisitos de la voz sobre IP.

Es un protocolo que define los códecs a utilizar, tanto en video como en audio, y los protocolos de transporte en la información. Fue el primero en utilizar el protocolo RTP.

- SIP: es un protocolo desarrollado por el IETF en 1999 para el control de llamadas multimedia y la implementación de servicios telefónicos avanzados. Está basado en HTTP adoptando la sencillez de su sintaxis y una estructura cliente/servidor basada en un modelo de petición/respuesta.

  Uno de los puntos fuertes de SIP es que da servicio tanto en TCP, puerto 5060, como en UDP para poder conectar a los clientes con el servidor SIP, pudiendo así obtener mayor o menor fiabilidad según lo que se necesite. SIP se utiliza únicamente para iniciar y terminar llamadas de voz y video. La comunicación de voz/video va sobre RTP.

- IAX: Inter-Asterisk eXchange protocol (IAX) fue desarrollado por Digium para la comunicación entre centralitas basadas en Asterisk aunque actualmente se ha implementado clientes que también soportan este protocolo.

  Éste protocolo consigue minimizar el ancho de banda utilizado en la transmisión de voz y video a través de la red IP. Su estructura básica se fundamenta en la multiplexación de la señalización y del flujo de datos sobre un puerto UDP, el 4569.

- SKINNY: es un protocolo desarrollado originariamente por Selsius y que actualmente pertenece a Cisco Systems. Es un protocolo que permite una comunicación eficiente con un sistema Cisco Call Manager. El Gestor de Llamadas actúa como un proxy de señalización para llamadas iniciadas a través de otros protocolos como SIP y H.323.

  El cliente se conecta a través de TCP con los Call Manager y para el tráfico de datos utiliza RTP/UDP/IP.

Como podemos ver, cada uno de los protocolos tiene sus propias peculiaridades y características, lo que hace indispensable disponer de un disector particular para cada uno. Con ello, podremos obtener diferentes tipos de información sobre la conexión según las limitaciones y los aspectos distintivos de cada protocolo de señalización.

2.4 Protocolo de señalización UNISTIM

En éste apartado se explicará cómo funciona el protocolo de señalización de VoIP UNISTIM. Se enumerarán los tipos de mensaje que dispone, además de explicar el esquema de flujo necesario de mensajes de señalización para establecer una llamada entre dos terminales.

El protocolo UNISTIM [6] fue creado por Nortel y adquirido por Avaya posteriormente. Es usado para la comunicación entre teléfonos IP o Softphones. Es un protocolo que se basa en el principio de maestro/esclavo, donde el esclavo informa de todas las acciones que se realicen con el teléfono IP y el maestro se encarga de dar órdenes al esclavo, mostrándole las posibilidades de uso del teléfono según en el estado en el que se encuentre.

Se puede decir que es un protocolo de estímulos, debido a que el esclavo se encarga de informar sobre el estímulo que le ha llegado al maestro, y según sea este estímulo, el maestro da una serie de posibilidades y de tareas. Dichos estímulos pueden ser implementados de manera sencilla en los terminales sin modificar el software en exceso. Esto hace que sea más sencillo de realizar el mantenimiento y la actualización de los terminales.

El hecho de ser un protocolo de estímulos hace que difiera de protocolos funcionales como SIP o H.323, que imponen que el servicio sea definido en un estándar y que los terminales dispongan de la lógica necesaria correspondiente a ese protocolo. Esto permite a los fabricantes y desarrolladores una mayor flexibilidad.

Entrando ya en los paquetes de UNISTIM, vemos que funciona bajo las capas de UDP/IP. Al estar bajo UDP utiliza un transporte no fiable, ya que no soporta confirmaciones de llegada de los paquetes, ni tampoco ningún tipo de control de flujo. Para ello UNISTIM tiene una sub-capa llamada *Reliable* UDP (UDP fiable) dentro de sí mismo, que consiste un número de secuencia y un indicador que nos dice el tipo de paquete que es, pudiendo ser Payload o ACK. Cuando uno de los extremos manda un mensaje de tipo Payload con un número de secuencia, el otro extremo le responde un ACK con el número de secuencia del paquete que le llegó. Con éste método se consigue otorgar un poco más de fiabilidad al enlace más allá del simple UDP y así saber si llegó el mensaje. Por último, antes del cuerpo UNISTIM, se añade un agregado UNISTIM que nos indica si viene por parte del terminal o del *Call Manager*. En caso de ser del terminal también nos indica el ID del terminal. Con ésta sub-capa conseguimos diferenciar e identificar la procedencia del mensaje, es decir, si es del *Call Manager* o del terminal.

Entrando ya en el cuerpo de la cabecera UNISTIM vemos que los mensajes se dividen en dos tipos, los enviados por el terminal y los enviados por Call Manager:

- Enviados por el terminal:

  - ✓ Key/ Indicator Manager Phone: Es un mensaje mandado hacia el Gestor de Llamadas con el fin de informar que tecla ha marcado el usuario en el terminal, tal y como se ve en la Figura II.

  - ✓ Audio Manager Phone: Tipo de mensaje enviado por el terminal para indicar en qué estado se encuentra el flujo de audio. También se usa para mandar estadísticas RTCP una vez se ha realizado la llamada.

- ✓ ACK: sirve para informar al otro extremo de que el mensaje llegó de manera correcta, enviando el número de secuencia del mensaje recibido dado por la sub-capa Riable UDP.

- Enviados por el Gestor de Llamadas:

  - ✓ Audio Manager Switch: Mensajes utilizados para ordenar tareas sobre el flujo de audio a los terminales. Tales como, ordenar que de tono de llamada, que conecte el transductor y que ajuste el volumen de la llamada. Con éste tipo de mensajes se controla el flujo de audio de la llamada, pudiendo abrirlo y cerrarlo.

    Dentro de éste tipo de mensajes se encuentran los que ordenan los parámetros a configurar por el terminal para que se realice la llamada. Tales como el codificador de voz a usar, puertos para el tráfico RTP y RTCP y la dirección IP del otro extremo en caso de ser necesaria, tal y como se muestra en el ejemplo de la Figura III.

  - ✓ Display Manager Switch: Son el tipo de mensajes que se encargan de la parte visual del terminal. Con éste tipo de mensaje se ordena al terminal los mensajes que debe mostrar por pantalla, llegando incluso a poder actualizar la interfaz del terminal mostrando así, que funciones están disponibles en cada caso. Los mensajes pueden ser tan dispares como desde mostrar la hora o el número del usuario, en qué estado se encuentra la conexión y el identificador de la persona que está llamando.

    Hay una parte de éstos mensajes que son personalizables por el Gestor de Llamadas, es decir, es la centralita la que elige qué mostrar por pantalla de manera personalizada.

  - ✓ Key/ Indicator Manager Switch: Estos mensajes se encargan de actualizar los elementos de hardware del terminal que ayudan a tener una referencia visual sobre llamadas o estado del terminal. Es el caso de encender LEDs para indicar que está encendido el manos libres o que la llamada está en curso, entre otras posibles aplicaciones.

  - ✓ ACK: sirve para informar al otro extremo de que el mensaje llegó de manera correcta, enviando el número de secuencia del mensaje recibido dado por la sub-capa *Riable* UDP.

Cada uno de esos tipos, salvo el ACK que se manda como paquete único, se cataloga como CMD dentro del paquete UNISTIM. Dentro de un mismo paquete puede haber varios CMD, es decir, que el tamaño de los paquetes es variable dependiendo del número de CMD que se manden en un mismo encapsulado. Por ello, dentro de cada uno de ellos se indica la longitud de cada uno para no mezclar contenidos de unos y de otros. Ésta información también nos ayuda para saber si el mensaje llegó defectuoso en longitud, es decir, si se perdió información.

Para poder diferenciar el tipo de mensaje se utilizan códigos que ocupan un byte. Cada código es único para cada mensaje y cada mensaje tiene una estructura predefinida según el sub-tipo que tenga. Es decir, conociendo el tipo de mensaje, el sub-tipo y la longitud podemos situar los campos y, obtener así, los códigos que nos dan la información del mensaje.

A continuación se muestran dos ejemplos de capturas realizadas con Wireshark:

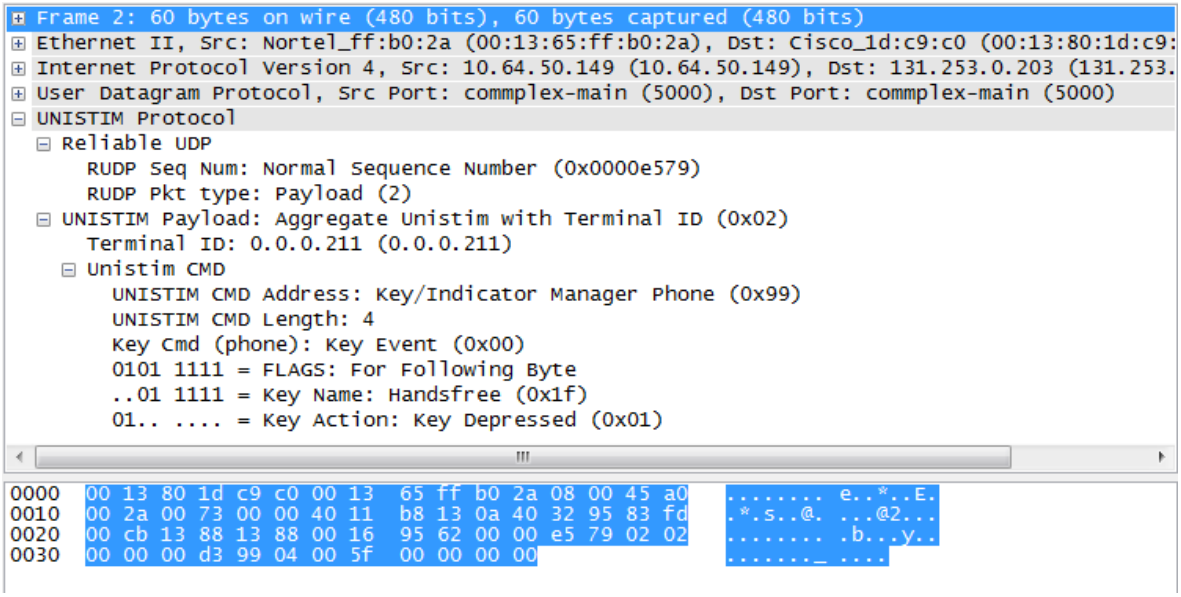

Figura II : Captura paquete del Terminal

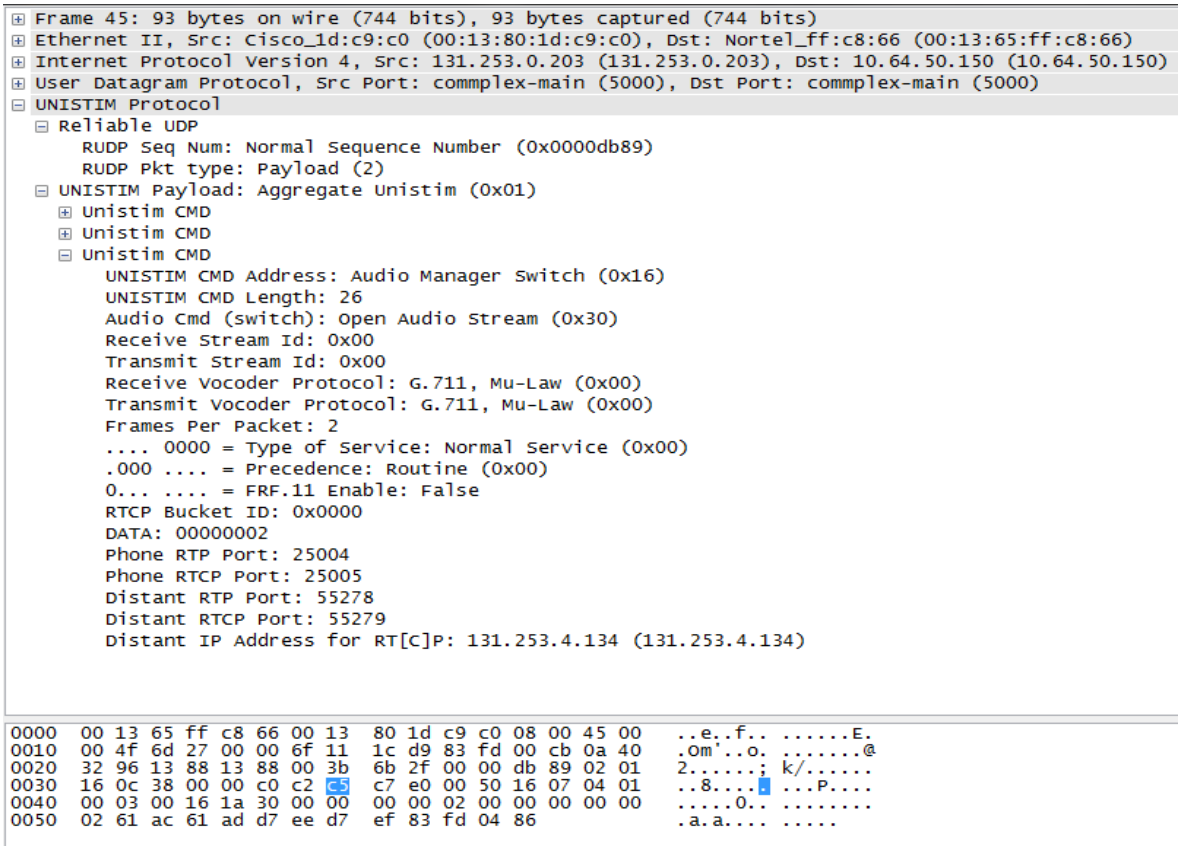

Figura III : Captura parámetros Gestor de Llamada

Una vez visto el tipo de mensajes y su estructura, pasamos a ver lo que sería el flujo de paquetes de señalización de UNISTIM necesarios para realizar una llamada entre dos terminales [6]:

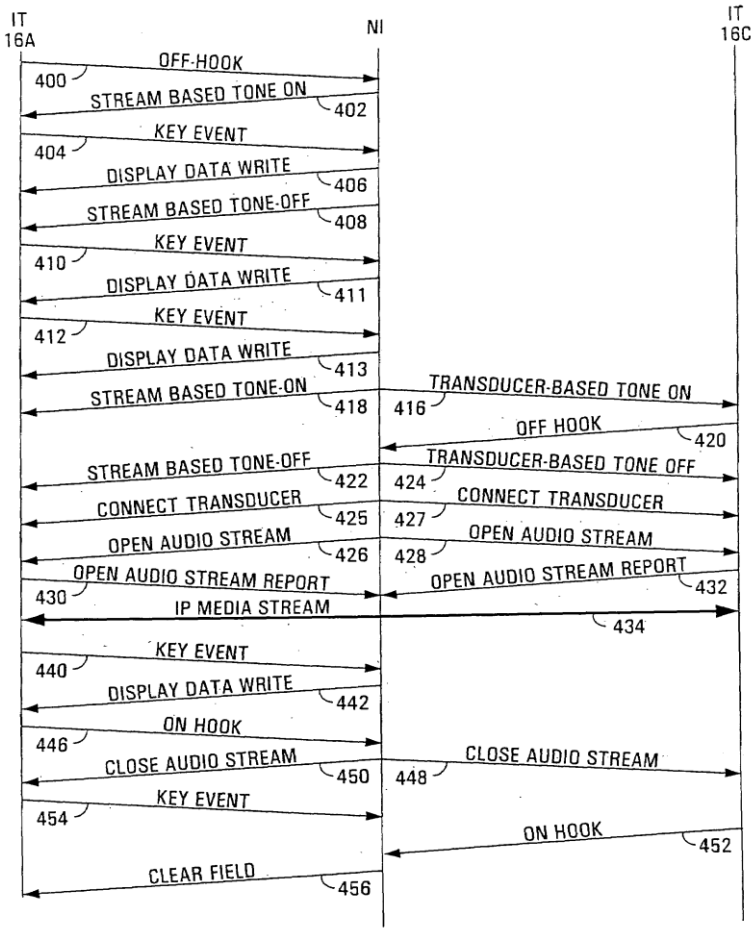

Figura IV : Esquema flujo UNISTIM [6]

Lo primero que debe ocurrir en una llamada es que se descuelgue el teléfono del llamante. Una vez se descuelga, se manda un mensaje (400), para indicarlo, hacia el Gestor de Llamadas. En el caso de los Softphones, el mensaje varía ya que no se puede descolgar como tal, sino que se indica mediante la utilización de los auriculares o la funcionalidad de manos libres. Acto seguido, la centralita ordena que de tono de llamada al terminal (402) para que el usuario que realiza la llamada sepa que la línea está operativa. Éste tono dura hasta que se marca alguna de las teclas de llamada (404) y se indica con un mensaje de que se termine el tono (408). En cada tecla que se marca, el Gestor le responde con un mensaje del tipo Display Manager Switch (406, 411, 413), indicando por pantalla que teclas se están marcando y en qué estado se encuentra la llamada. Una vez se marque el número a llamar, es el Gestor el que se encarga de comprobar si existe ese número y ver si se encuentra disponible.

En caso afirmativo, la centralita hará que el emisor reciba tono de llamada (418) y el terminal del receptor comience a sonar (416). Una vez el receptor descuelgue la llamada (420) se informará a la centralita para que establezca el flujo RTP. Para ello el Call Manager lo primero que hace es ordenar que termine el tono de llamada en ambos terminales (422, 424). Posteriormente envía órdenes a ambos terminales, indicando qué

parámetros deben de tomar para abrir el flujo de audio (426, 428) una vez se conectaron los transductores de ambos extremos (425, 427).

Antes de abrirse el flujo RTP, se necesita que esté abierto el Audio Stream. Los terminales indican que está abierto respondiendo al mensaje que indicaba los parámetros de configuración de una manera satisfactoria (430, 432). A partir de ese momento es posible que se realice el flujo RTP (434).

Para finalizar la llamada, alguno de los dos terminales pulsa la tecla de colgar, y como todas las teclas que se marcan, se indica al Gestor (440). Posteriormente el usuario cuelga el teléfono de manera física (446), al igual que en el caso de descolgar, en un Softphone éste paso no se realiza. Una vez enterado el Gestor, es éste el que ordena que cierre el flujo de audio en ambos extremos (448, 450).

Ya por último, el receptor cuelga físicamente el teléfono (452) y se ordena limpiar a ambos terminales los parámetros utilizados y los mensajes mostrados por pantalla (456).

Una vez visto el flujo de mensajes necesarios para establecerse la llamada con el protocolo UNISTIM, cabe recalcar que el orden de los mensajes depende del Gestor que se use.

2.5 Proyecto VoIPCallMon:

En este apartado se expondrán las razones de la existencia del proyecto VoIPCallMon, cuál es su finalidad y su funcionalidad. Además se explicará cómo funciona y ver así dónde se introduce el módulo de UNISTIM.

El proyecto VoIPCallMon nace para intentar resolver los problemas existentes en la monitorización, como son la capacidad de monitorizar a velocidades altas y la de guardar ciertos datos de las llamadas, al igual que se hace en la telefonía actual. Se plantea como una nueva tecnología capaz de reducir costes en la monitorización a gran escala. En las grandes empresas que dan servicio de VoIP, es necesario tener varios puntos de monitorización de la red y esto conlleva una gran inversión. VoIPCallMon se crea para poder hacer una monitorización mucho menos costosa, con un Hardware de menos coste, eso sí, teniendo un software optimizado.

A continuación se explicará de manera esquemática como se optimiza el software VoIPCallMon para la monitorización. Esto nos ayudará a conocer donde se introduce el módulo UNISTIM y de qué forma.

El software está escrito en lenguaje C en un sistema basado en Linux. Se divide en cuatro módulos bien diferenciados. Son los siguientes [7]:

- **Módulo de captura del tráfico**: Es el módulo encargado de capturar el flujo de tráfico. Es capaz de capturar a velocidades entorno a los Gb/s usando un

Hardware disponible en el mercado y dando soluciones de software a sus limitaciones.

- **Módulo de detección y rastreo de tráfico del protocolo de VoIP**: Es el módulo encargado de realizar la detección y la búsqueda del tráfico del protocolo de VoIP y del RTP. Es decir, en nuestro caso es el módulo de UNISTIM. Una vez se rastrea el tráfico, nuestra solución se encargaría de realizar la disección y monitorización. Por lo tanto, nuestra solución se insertaría en este módulo.

- **Generador de grabaciones y llamadas de VoIP**: Una vez la llamada se haya terminado y se haya eliminado del módulo anterior, entra en funcionamiento éste módulo. Es el encargado de generar toda la información requerida sobre la llamada que expiró. La información proviene del módulo anterior, pero éste es el módulo que se encarga de tratarla y extraerla.

- **Modulo de retención de llamadas VoIP:** Periódicamente, la salida del módulo anterior es volcada a una base de datos MySQL. Con el fin de que, clientes, operadores, administradores de la red o cualquier otro agente puedan recuperar llamadas, medidas o alarmas, teniendo la identificación oportuna.

2.6 Conclusiones

En este capítulo se ha mostrado el estado del arte en los protocolos de señalización de VoIP. Se ha hecho especial hincapié en detallar el protocolo que se pretende diseccionar y se ha hecho un seguimiento de cómo funciona el flujo de mensajes. Además se ha explicado cómo funciona el proyecto en el que se tiene que introducir el módulo de UNISTIM, VoIPCallMon. En los siguientes apartados se usarán estos conocimientos con el fin de diseccionar de una manera eficaz el protocolo y de cómo se tiene que configurar para adaptarse al proyecto.

Además se ha podido ver que UNISTIM es un protocolo claramente definido del tipo maestro/esclavo, ya que es el Gestor, el encargado de dirigir toda la llamada. Mientras que la tarea del terminal es mantener informado a la centralita de las teclas que se van marcando y de si se consigue, abrir de manera correcta el flujo de audio. Es decir, no mantiene ni gestiona ningún tipo información más allá de la proporcionada por el Gestor de Llamadas.

En el próximo capítulo se expondrán los requisitos que tendrán que tenerse en cuenta en la realización de la aplicación. Además, se expondrán de una manera más técnica y de forma numerada para que quede más claro.

3 Análisis del problema

3.1 Introducción

En éste tercer capítulo se aborda qué tipo de aplicación necesitamos para llevar a cabo éste TFG. Mediante la enumeración de los objetivos se dará una visión más concreta sobre el problema al que nos enfrentamos y de manera complementaria, se podrá ver qué aspectos son primordiales para llegar a ello.

El modelo que se expone en éste documento afecta únicamente a la parte del protocolo UNISTIM, incluyendo la adaptación al proyecto VoIPCallMon y todo lo que esto conlleva. Es decir, disecciona toda la parte de las capas UDP y UNISTIM que interesen, para poder así realizar un seguimiento de las llamadas que se produzcan en el punto de monitorización.

De ésta forma se podrá aplicar el módulo de UNISTIM dentro de VoIPCallMon para diseccionar de manera rápida y eficaz, cumpliendo así éstos requisitos del proyecto VoIPCallMon.

3.2 Análisis de requisitos

A continuación se mostrará una lista de los requisitos que nos imponemos para realizar la aplicación. Se hará una división entre requisitos funcionales y no funcionales, para que se vea de manera clara, cuales son los aspectos que se abarcan en nuestra solución.

3.2.1 Requisitos funcionales

En esta sección se explicarán las metas funcionales que buscaremos en nuestra aplicación, es decir, de lo será capaz nuestra aplicación de llevar a cabo. Para que sea más fácil de comprender se dividirán los requisitos en módulos. Cada módulo se encarga de realizar una tarea dentro de la monitorización global. Todas ellos funcionan como una cadena de procesado y actúa según se les necesite. Los módulos son:

- Captura del tráfico: Ésta primera parte será la encargada de recoger el tráfico que circula en el punto de monitorización. Una vez capturado también tendrá que ser capaz de extraer los datos de interés, en nuestro caso, los mensajes de tipo UNISTIM. También deberá extraer RTP en nuestro caso cuando sea pedido por la aplicación.

- Análisis UNISTIM: Ésta parte continuará el trabajo de la anterior. Una vez ya se haya extraído el tráfico UNISTIM, se hará un seguimiento de la conexión entre dos o más terminales mediante la información extraída y siguiendo la lógica implementada en la aplicación.

- Extracción de datos de la llamada: La tercera parte se encargará de extraer en documentos de texto toda la información RTP y UNISTIM de la conexión. Además extraerá todos los paquetes UNISTIM monitorizados y las grabaciones de voz de la llamada entre terminales.

A continuación se detallarán los requisitos funcionales de cada uno de los módulos de una forma técnica.

Captura del tráfico:

RF1: Deberá ser capaz de capturar todo el tráfico que circule para su posterior uso.

RF2: Se deberá implementar un filtro para obtener solo la información que nos resulte útil.

RF2.1: Un filtro para paquetes UNISTIM.

RF2.2: Un filtro para paquetes RTP.

RF2.2: Una vez filtrados los paquetes UNISTIM deberá ser diseccionado más a fondo para obtener toda la información de señalización que nos brinda éste protocolo.

Análisis del tráfico UNISTIM:

RF3: Se implementará un gestor capaz de saber en qué estado se encuentra la llamada gracias a los datos recogidos.

RF3.1: Dispondrá de un máquina de estados para saber en qué situación está. Actualizará el estado según las normas de diseño que implementemos. La aplicación tiene que ser capaz de seguir esa máquina de estados sin error.

RF4: Se implementará un gestor capaz de clasificar las llamadas.

RF4.1: Tendrá una tabla clasificatoria de las llamadas producidas para poder actualizar el estado de las conexiones.

RF4.2: Podrá introducir o eliminar llamadas de la tabla según criterios del diseño.

RF5: Se deberán obtener todos los parámetros de configuración de la llamada que nos brinda UNISTIM.

RF5.1: Obtendrá las IPs de ambos puntos, además de los puertos a utilizar, tanto para RTP como RTCP, como el codificador de voz utilizado en la conexión.

RF6: Deberá saber qué mensajes provienen del terminal y cuál del Call Manager.

RF6.1: En el caso de que el tráfico RTP involucre a otro terminal y no se produzca a través del Call Manager, deberá ser capaz de identificarlo y de monitorizarlo de manera adecuada.

RF7: Se creará un gestor del flujo RTP capaz de saber cuándo clasificar el flujo y cuando estar inactivo.

RF7.1: Solo se activará cuando se llegue a producir la llamada de manera satisfactoria y con los parámetros correctos.

Extracción de datos de la llamada:

RF8: Se creará un gestor capaz de extraer datos de las llamadas solo cuando nosotros hayamos estipulado.

RF9: Se almacenarán los datos obtenidos de la llamada en un sistema de ficheros.

RF9.1: Se extraerán parámetros que han sido necesarios para la configuración el flujo RTP.

RF9.2: Se extraerá información de interés del flujo UNISTIM.

RF10: Será capaz de extraer el flujo de voz en ambos sentidos de la llamada en ficheros diferentes y con la posibilidad de combinarlos.

3.2.2 Requisitos no funcionales

Son otros aspectos que nuestra aplicación debe cumplir, pero que no se contemplan como objetivos funcionales. Deben cumplirse para dar una mayor robustez a nuestra aplicación. En nuestro caso buscaremos:

RNF1: Rendimiento: Debe tener un alto rendimiento para poder monitorizar en tiempo real y para poderse utilizar en cualquier dispositivo comercial con mínima capacidad.

RNF2: Estabilidad: La aplicación debe ser altamente estable, ya que podría utilizarse de manera continua y sin pausas en caso de monitorizar en tiempo real.

RNF3: Disponibilidad: Debe estar disponible cuando se requiera para poder monitorizar tráfico en cualquier momento y durante el tiempo que sea necesario.

RNF4: Modificabilidad: El código debe estar lo más esquematizado posible para poder realizar modificaciones cuando se requiera.

RNF5: Gestionabilidad: El código debe estar correctamente documentado y explicado.

RNF6: Portabilidad: Debe ser una aplicación capaz de poderse trasladar a otros dispositivos o puntos de monitorización.

RNF7: Seguridad: Tanto la aplicación como los datos deben estar en un lugar protegido mediante autenticación.

RNF8: Sencillez: Los datos extraídos en los resultados deben ser fáciles de comprender.

3.3 Conclusiones:

En este capítulo hemos visto como queremos que sea nuestra aplicación. Lo hemos dividido en partes para poder así, atajar los problemas de manera más específica y más comprensible. Además buscando siempre un buen rendimiento para poder trabajar a tiempo real. También es primordial el tener un código bien estructurado, para poder hacer un mantenimiento a medida que pasa el tiempo y una sencilla obtención de resultados, para que sea fácil de interpretar.

En los capítulos próximos irá tomando forma la aplicación y se verán que soluciones se han tomado a todos estos requisitos.

4 Desarrollo de la solución

4.1 Introducción:

En este capítulo se mostrará cómo se ha desarrollado la aplicación para llegar a cumplir todos los requisitos mencionados en el anterior capítulo. Para ello, se seguirá con la idea de división en módulos comentada en el anterior capítulo. Así se dispondrá de una explicación más estructurada y comprensible. Cada módulo tiene sus propias tareas. Los módulos son dependientes entre sí por lo que se hará una explicación según el siguiente orden:

- Captura del tráfico.
- Análisis del tráfico UNISTIM.
- Extracción de resultados.

Además se introducirá una tarea nueva, la adaptación al proyecto VoIPCallMon:

- Integración del módulo UNISTIM en VoIPCallMon.

4.1 Captura del tráfico

En este sub-apartado se explicará cómo se diseccionan los paquetes que se capturen del protocolo UNISTIM. Se partirá del trabajo realizado por VoIPCallMon, que realiza el filtrado hasta la capa IP dejándonos la disección de UDP y UNISTIM a nosotros. Cabe recalcar que la disección que se presentará a continuación se hace paquete por paquete.

Realizaremos una explicación lineal a lo largo de la disección para que se vea más algo más claro:

Fase 1: Una vez VoIPCallMon identifica que el paquete es IP el siguiente paso a realizar es averiguar si el paquete pertenece al protocolo UNISTIM. Para saber si pertenece o no se comprueban que puertos se están utilizando. En el caso de UNISTIM, los puertos utilizados son el 5000 y el 5100. Si se diera el caso en el que se usan esos puertos significaría que es un paquete UNISTIM.

A continuación, se recogerían los siguientes datos del paquete: longitud del paquete, puerto destino, puerto fuente, IP destino, IP origen, puntero a la posición de memoria donde comienza el Payload.

Con toda esa información pasaríamos a la disección UNISTIM.

Fase 2: Ya tenemos todo para comenzar a diseccionar el paquete UNISTIM. Lo primero, se comprueba si el paquete que nos llegó pertenece a una conexión existente. En caso afirmativo, se le cargaría la información que se disponga de esa conexión. Si no es así, se introduce en la tabla de conexiones.

Fase 3: Una vez comprobado si se encuentra en la tabla, avanzamos el puntero para ver qué tipo de paquete UNISTIM es. Si es un paquete ACK, lo contabilizamos y acabamos la disección ya que no se obtiene más información. Mientras que si es un tipo *Payload* continuaríamos con la disección. Esto es porque los paquetes *Payload* contienen información relevante sobre el estado de la conexión.

Fase 4: Ahora, aprovechando que el protocolo UNISTIM nos brinda la información de si un paquete proviene del Gestor de Llamadas o del Terminal, hacemos la comprobación. Esto nos da dos vertientes a diseccionar, por un lado en el caso de que sea del Terminal, y por otro, del *Call Manager*.

Fase 4.1: Si detectamos que proviene del Terminal lo primero que hacemos es obtener el identificador. Una vez obtenido el identificador se pasa a diseccionar el cuerpo del paquete UNISTIM, los mensajes que contiene en su interior. Se recuerda que cada mensaje, o CMD, que es como lo nombramos en el capítulo 2, dispone de un campo de identificación del tipo de mensaje y otro campo para indicar la longitud del CMD. Con ayuda de estos campos avanzaremos por cada CMD extrayendo la información que nos resulte de interés.

Llegados a este punto, lo primero que tenemos que hacer es obtener el tipo de CMD y su longitud. Una vez extraído comprobamos si es de los siguientes tipos:

- *Key Manager Phone:* Al ser de este tipo podremos obtener que tecla es la que se ha pulsado. Se puede ver un ejemplo en la Figura II.

- *Audio Manager Phone:* Este tipo de mensajes son aquellos que comunican al Gestor de Llamadas en qué estado se encuentra el flujo de sonido. Es decir, si se consiguió abrir o no el flujo.

Fase 4.2: Si el mensaje proviene del Gestor de Llamadas, se tiene que hacer una disección diferente. La primera diferencia es que el Gestor no dispone de identificador como tal. Por lo tanto, lo primero que nos encontramos es el tipo de CMD y la longitud. Llegados a este punto la disección es similar a la del Terminal, es decir, tenemos que obtener también el CMD y la longitud. También existe la posibilidad de que existan varios CMD dentro de un mensaje UNISTIM. Así que también tiene que tenerse en cuenta para diseccionar.

Una vez extraídos CMD y longitud comprobamos de qué tipos son. Los tipos que nos interesan en el caso de ser provenientes del Gestor son:

- *Audio Manager Switch*: En este tipo de mensajes se puede obtener información sobre órdenes que manda el Gestor de Llamadas al Terminal para hacer posible el flujo de audio. En nuestro caso, se comprueba si se ordena activar el transductor, ya que nos puede dar información sobre el estado en el que se encuentra la llamada.

- *Open Audio Stream:* Es la clase de mensaje que más información nos da sobre la conexión. Es el mensaje que manda el Gestor para configurar los parámetros del Terminal. Los parámetros de configuración que se obtienen en este mensaje se muestran en la Figura III, y son:

    - ✓ Puertos RTP a usar en el flujo de audio, por los dos extremos.
    - ✓ IP para el flujo RTP del otro extremo en el caso de que la información no viaje a través del Gestor.
    - ✓ Códec de voz que tiene que usar tanto un extremo como el otro.

  Por último, decir que UNISTIM tiene la posibilidad de establecer el flujo RTP de extremo a extremo o pasando por el Gestor de Llamadas. La centralita es la que toma esta decisión y también la encargada de comunicárselo a los terminales. En el caso de ser entre extremos, introduce un campo al final del mensaje *Open Audio Stream*, indicando la IP a utilizar para el RTP. Mientras que si tiene que pasar por el Gestor, simplemente no se le indicaría otra dirección. Este aspecto se tiene en cuenta a la hora de la disección con ayuda del campo de longitud del CMD.

- *Close Audio Stream:* Es el tipo de mensaje encargado de ordenar que se finalice el flujo de audio a cada terminal. Por lo tanto, es muy importante tenerlo en cuenta ya que nos dice cuando una conexión fue eliminada.

Así es como quedaría el esquema de disección del protocolo:

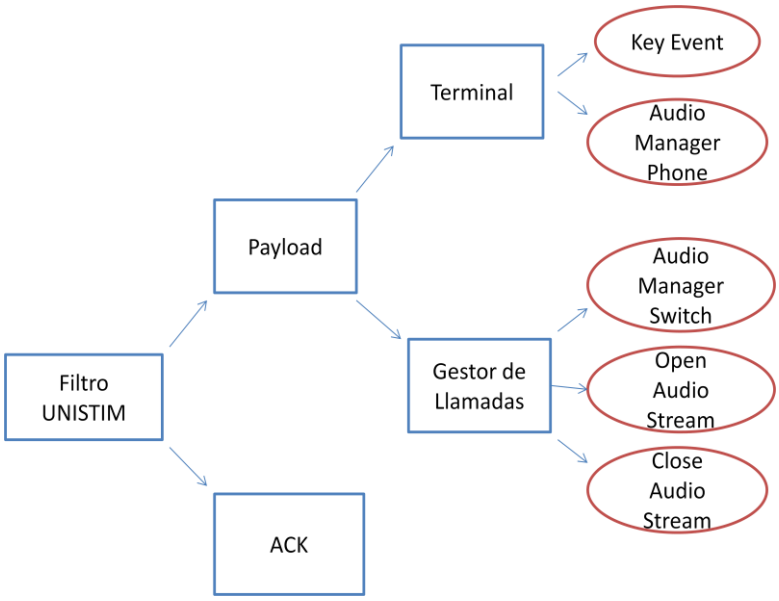

Figura V: Esquema de disección de UNISTIM

Por último, habría que decir, que dentro de cada CMD, en los tipos que nos interesan, se realiza una comprobación de si éste se encuentra dentro de una llamada registrada en la tabla de llamadas. En caso afirmativo, se vuelca toda la información que se disponga de la llamada por si hiciese falta en algún momento utilizarla. Además, según el tipo de CMD se pueden actualizar los campos de la llamada según la información que nos dé el mensaje.

4.2 Análisis del tráfico UNISTIM

Una vez se sabe de dónde obtener la información y cómo hacerlo, tenemos que avanzar al siguiente paso. Es decir, realizar el seguimiento de la llamada con los datos obtenidos y según el paquete que nos llegue en cada momento. Así se podrá saber cuándo obtener información relevante de la conexión y realizar una monitorización adecuada. También nos ayudará a saber cuándo debe empezar a fluir el RTP entre los dos extremos, pudiendo capturarlo para grabarlo.

Para realizar esta tarea se ha decidido seguir una lógica parecida a la de una máquina de estados. Según el mensaje que nos llegue, y en qué estado estemos, se podrá obtener información sobre el enlace. Esto resulta muy útil para saber si se realizó una llamada correctamente o pasó algo extraño.

A continuación se va a proceder a explicar el funcionamiento de la lógica que tiene la aplicación. Para que sea más entendible vamos a explicar dos casos. Los casos que se pueden dar en una llamada, monitorización donde se recibe la llamada o donde se emite. Además, dentro de cada caso, se deberá guardar el estado tanto del Terminal como del Gestor de Llamadas:

- **Llamada emitida:** La máquina de estados que se va a seguir es la siguiente:

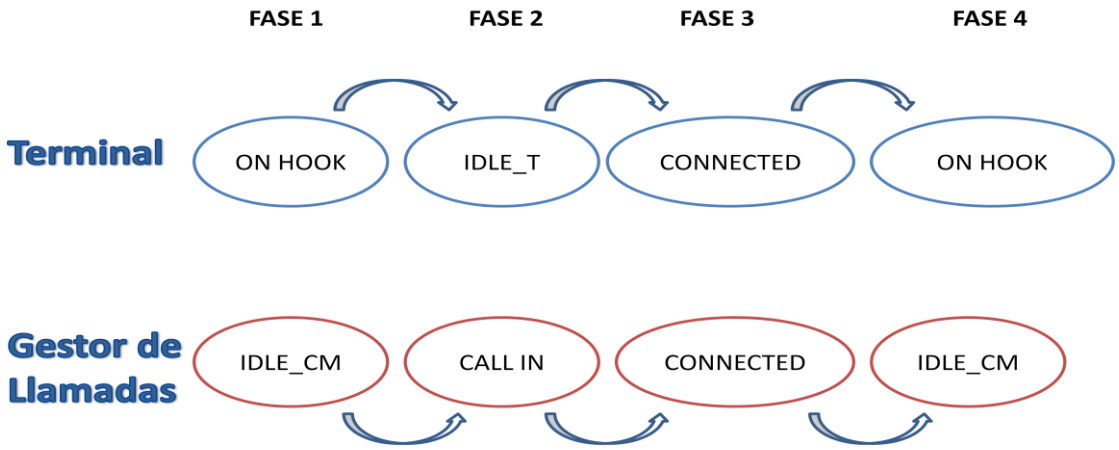

Figura VI: Máquina de estados llamada emitida

✓ Fase 1: Se encuentra el sistema en fase de reposo, donde el terminal se encuentra colgado y el Gestor de Llamadas activo, pero en espera de recibir algún estímulo. Para pasar a los siguientes estados se tiene que

capturar y monitorizar algún mensaje en el que se ha pulsado la tecla de llamar o cualquier tecla perteneciente a los números. Éste caso también se tiene que tener en cuenta, ya que en el caso de los Softphones puede no existir el botón de realizar llamada o tener la alternativa de realizar la llamada sin pulsarlo.

Una vez capturado el mensaje, pasa el terminal a modo de espera. El mensaje también hace que pase de *modo de espera* a modo de *llamada entrante.*

Durante esta fase se recopila la siguiente información:

a) Identificador del terminal.
b) Tomamos como inicio de la llamada en la parte que envía el paquete, el Terminal.
c) IP del que emite el paquete, es decir, el Terminal.
d) Puerto UNISTIM del Terminal.
e) IP del que recibe el paquete, es decir, el Gestor de Llamadas.
f) Puerto UNISTIM del Gestor.

- ✓ Fase 2: Durante esta fase el terminal espera a que se realice toda la marcación de números. Mientras que el Gestor de Llamadas recibe toda información para establecer la llamada. Una vez el Gestor verifique que el número es correcto y que no se encuentra ocupado o apagado, manda el mensaje que proporciona todos los parámetros de configuración al terminal. Estos parámetros de configuración se usan para crear el flujo de audio entre los dos extremos. Es decir, nos otorga la siguiente información:

  a) Puertos del flujo RTP de ambos extremos de la conexión.
  b) Puerto RTCP de ambos extremos.
  c) *Códecs* de audio que utilizarán ambos extremos.
  d) IP del otro extremo en caso de que la conexión no circule a través del Gestor de Llamadas. Se reitera que este aspecto lo configura la propia centralita.
  e) Tiempo de inicio de llamada en el otro extremo. Tomamos este paquete como inicio debido a que no se puede conocer cuando se ha conectado el transductor en el otro extremo en el momento exacto. Sin embargo, cuando se manda este paquete el transductor ya tiene que estar conectado.

Para poder pasar a la siguiente fase, el Terminal tiene que informar que ha configurado todos los parámetros de manera adecuada con un mensaje

del tipo *Audio Manager Phone* visto en el Capítulo 2. La siguiente fase, como se ve en la Figura VI, es cuando ambos están conectados.

- ✓ Fase 3: En ésta fase es cuando comienza a producirse todo el flujo de audio entre los dos extremos. Es decir, aquí es donde se comienza a capturar el flujo RTP con los parámetros obtenidos en la anterior fase. Con el flujo de audio seremos capaces almacenarlo y extraer sus parámetros más relevantes. La llamada se mantendrá en este estado hasta que se reciba un mensaje que haga cerrar la llamada.
- ✓ Fase 4: Es la última fase de la llamada. Se llega a este punto cuando se recibe un mensaje de cerrar el flujo de audio por parte del Gestor. El flujo puede ser cerrado por cualquiera de los dos extremos pulsando la tecla de colgar informando al Gestor, y es éste último el que lo ordena. Además la centralita puede cerrar la llamada cuando le resulte necesario, ya sea por problemas de conexión o cualquier otro factor. Es decir, es el Gestor el que tiene la última palabra, ya que tiene el poder de ordenar cerrar el flujo.

  En esta fase obtenemos la siguiente información:

  a) Tiempo de finalización de la llamada para ambos extremos.

- **Llamada recibida:** Al monitorizar donde se recibe la llamada, el camino que se realiza la máquina de estados es algo diferente:

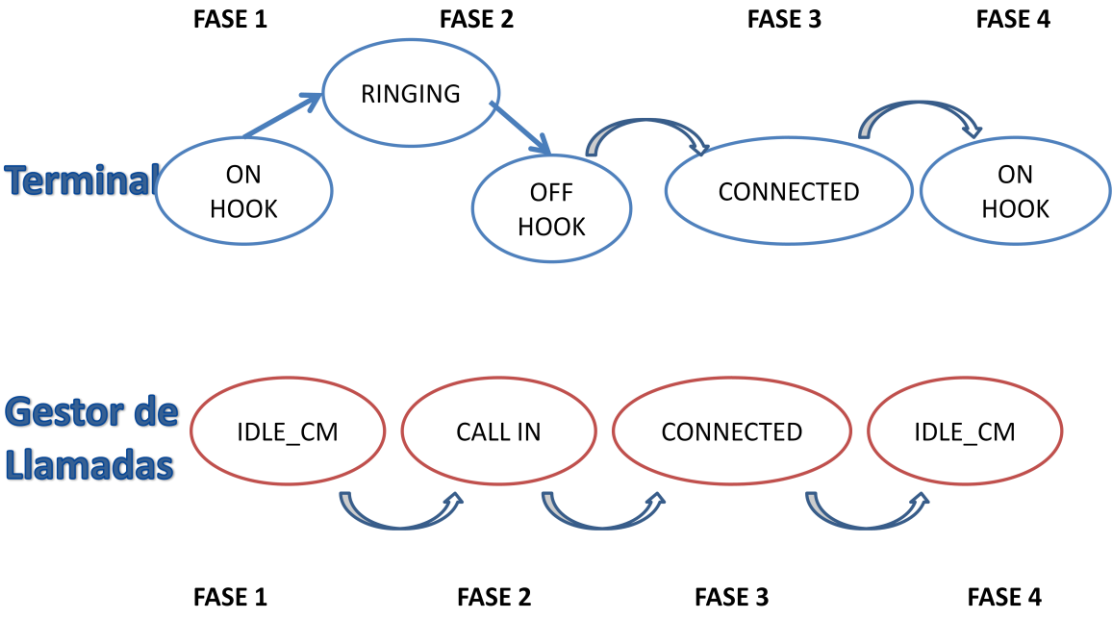

Figura VII: Máquina de estados llamada recibida

En el caso de que la llamada sea recibida, la fase que varía es la número 2 y el paso de la 1 a la 2 respecto al caso anterior. En el resto de casos, el

procedimiento es el mismo. Por lo tanto se explicarán únicamente esas diferencias a la hora de realizar el análisis.

Fase 2: Al recibir una llamada, el procedimiento es algo diferente. Para pasar del estado de reposo, o de estar colgado, al estado en el que el teléfono está siendo llamado, tiene que ocurrir que el Gestor lo informe al terminal. Para ello, lo informa mediante un mensaje pidiendo que conecte el transductor, *Transducer Based Tone On* del tipo *Audio Manager Switch*, visto en el capítulo 2. Una vez nos llega ese mensaje podemos decir que el teléfono está sonando, es decir, en el estado de *Ringing* como se ve en la Figura VII.

Nos mantendremos en este estado hasta que el terminal indique a la centralita que se ha pulsado el botón de descolgar. En este momento se pasaría al segundo estado de la fase 2 en el Terminal. Una vez en el estado de *Descolgado, OFF HOOK*, es cuando se recibiría la información de parte del Gestor con los parámetros a configurar en el flujo de audio, y a su vez la confirmación de que realizó de manera satisfactoria, tal y como se ve en la Figura IV.

Por último, antes de acabar éste apartado, hay que recordar que el Gestor de Llamadas puede finalizar la llamada en cualquier momento. Es él el que se encarga de hacer que la llamada sea satisfactoria y por ello, en el caso de que exista algún problema, puede terminar la conexión cuando lo requiera. Este es, por ejemplo, el caso en el que se produzca congestión en la línea o que alguno de los dos terminales se encuentre inhabilitado.

4.3 Extracción de datos de la llamada

Por último, solo quedaría la extracción de los datos y resultados de la llamada. Una vez se ha recorrido toda la lógica de procesado implementada para una llamada ya disponemos de la información necesaria para extraer información de monitorización.

La extracción puede ocurrir cuando acabe una llamada, pero también se tiene que tener en cuenta que hay llamadas que no finalizan de manera adecuada. Para ello se implementa un sistema de tiempo límite de expiración. De este tema se profundizará en el capítulo siguiente, en la integración dentro de VoIPCallMon.

Los ficheros que se extraen son los siguientes:

- **Fichero de parámetros RTP**: Es el fichero donde se encuentra toda la información relacionada con el flujo RTP, es decir, del flujo de audio. Dentro de cada fichero se guarda la información de todos los enlaces que se detectaron en el punto donde se monitorizó. Se extrae la siguiente información:
  - ✓ Direcciones IP de origen y destino.
  - ✓ Puertos RTP de origen y destino.

- ✓ Tiempo en el que se inició el flujo de audio y tiempo de finalización.
- ✓ Se especifica donde se almacenó la grabación del flujo. Para ambos sentidos, es decir, el audio de ida y el de vuelta.

- **Fichero de parámetros UNISTIM**: Es el fichero donde se puede encontrar toda la información de los enlaces sobre el protocolo UNISTIM. Es decir, es información relativa a la señalización entre Terminal y Gestor de Llamadas. Un ejemplo de fichero es el siguiente:

```
1 1400856071696036    1400856097300002    0.0.0.19    5000    5000    150.244.67.54   150.244.64.2    25.603966   23.609502   9
  1   ../Resultados/RAW//150.244.64.2-150.244.64.2-1400856071696036.raw   ../Resultados/
  RAW//150.244.67.54-150.244.67.54-1400856073690500.raw   ../Resultados/RAW//150.244.67.54-1400856071696036.unistim.pcap
  NO_AVAILABLE    NO_AVAILABLE    2002
2 1400856082933184    1400856097297958    0.0.0.19    5000    5000    150.244.67.61   150.244.64.2    14.364774   14.363701   9
  1   ../Resultados/RAW//150.244.67.61-150.244.67.61-1400856082933184.raw ../Resultados/
  RAW//150.244.64.2-150.244.64.2-1400856082934257.raw ../Resultados/RAW//150.244.64.2-1400856082933184.unistim.pcap
  NO_AVAILABLE    NO_AVAILABLE
```

Figura VIII: Ejemplo fichero de información UNISTIM

Como se puede ver, en este ejemplo, se han detectado seis flujos diferentes de tráfico UNISTIM. De cada uno se extrae:

- ✓ Tiempos de inicio y finalización de la llamada.
- ✓ Identificador del terminal.
- ✓ Puertos UNISTIM de origen y destino.
- ✓ IPs de origen y destino.
- ✓ Duración de la llamada en el emisor y en el receptor.
- ✓ Estado en el que acabó el Terminal y el Gestor de llamadas expresado numéricamente. En este caso, el 9 se corresponde con *On Hook* del Terminal y el 1 con *Idle* del Gestor.
- ✓ Indicación de en dónde se encuentra el flujo de audio, para ambos sentidos.
- ✓ Indicación de en dónde se encuentra el tráfico UNISTIM filtrado para esa llamada.
- ✓ Información de si disponía VLAN en el Terminal o en el Gestor.
- ✓ Teclas marcadas al realizar la llamada.

- **Ficheros con las capturas de los enlaces:** Son ficheros de tipo *pcap* donde se han filtrado todos los paquetes procedentes del protocolo de señalización para un enlace. Esta filtración nos permite ver de manera individual el protocolo UNISTIM para cada uno de los enlaces.

- **Grabaciones de voz:** Además también se almacenan las grabaciones de voz de ambos sentidos. Para cada enlace se crea una carpeta individual donde se almacena el flujo de voz que se produjo.

4.4 Integración del módulo UNISTIM en VoIPCallMon:

Una vez visto como funciona la aplicación y las soluciones tomadas para cada uno de los problemas vistos, ya dispondríamos del cuerpo del módulo UNISTIM. En este aparado se busca avanzar un paso más. Esto es, introducir el módulo creado, dentro del proyecto de disección del grupo de investigación *High Performance Computing and Networking* de la Universidad Autónoma de Madrid.

Nuestro módulo de UNISTIM se integraría una vez pasado el primer módulo según lo visto en el Capítulo 2. Como se mencionó en el apartado anterior, el primer módulo es el encargado de capturar todo el tráfico en el punto deseado para monitorizar. Una vez tenemos el tráfico disponible, es nuestro módulo el encargado de rastrear los mensajes del protocolo UNISTIM para su posterior análisis y procesado.

Para poder introducir el módulo nuevo en el proyecto, se tiene que seguir una estructura similar a la que se muestra a continuación [7]:

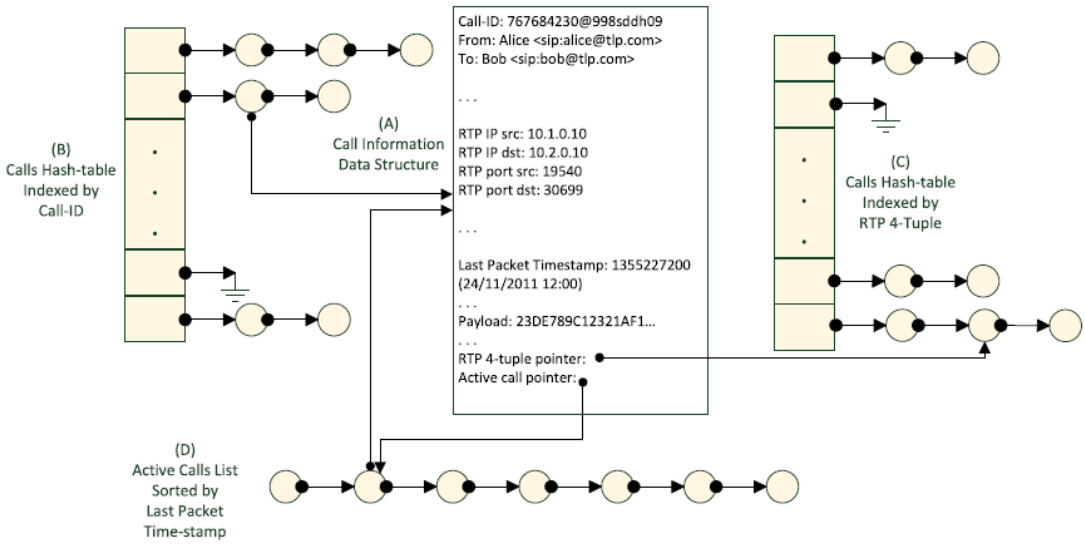

Figura IX: Estructura de datos para almacenar y mantener información de las llamadas en el módulo de detección de SIP/RTP [7]

La Figura IX muestra la estructura que sigue el segundo módulo para el protocolo SIP. En el caso de nuestro protocolo las estructuras a seguir son idénticas, por lo que se tomará esta figura como referencia.

Como se ve en la Figura IX, el módulo dispone de dos tablas Hash. Una indexada con el identificador de llamada y la otra con la cuádrupla de RTP. En el caso de UNISTIM, no

dispone de un identificador de llamada como tal, por lo que se decidió combinar las IPs de origen y destino, y los puertos de origen y destino, para asegurar que es único de ese enlace. Por otro lado, como se pueden producir colisiones, se usan listas con nodos unidos en cada entrada en cada una de las tablas hash para resolver esto. Los nodos y los enlaces están representados en la figura con círculos y flechas. Ambas tablas hash acceden a la misma estructura de datos mediante punteros. Esta estructura contiene información sobre la llamada y sobre su correspondiente flujo RTP.

La tabla de indexación de llamadas es útil para saber si una llamada ya se encuentra registrada, y si es así, actualizarla con el nuevo paquete UNISTIM que viene. Mientras que la tabla RTP es útil cuando es procesado un paquete RTP.

En las primeras etapas del sistema, una llamada de UNISTIM se marca como expirada cuando se recibe un mensaje de cierre por parte del Gestor de Llamadas, tal y como se explicó en el capítulo 2. Sin embargo, puede ocurrir que ese mensaje nunca llegue debido a algún fallo de la conexión. Esto hace necesaria que exista un tiempo de expiración, para que las llamadas que no se finalizaron de manera correcta, no ocupen memoria. Periódicamente se chequearían las tablas para comprobar el tiempo de expiración. Como este procedimiento tendría un coste computacional alto, lo que se hace es crear una lista de llamadas activas. Los nodos de esta lista tienen punteros a las estructuras que representan, donde dentro de esas estructuras tienen a su vez otros punteros a las entradas correspondientes de las tablas hash, tanto UNISTIM como RTP.

Por último, toda la memoria usada en la aplicación, como estructuras, nodos, listas y tablas hash, está reservada en una fuente de memoria para reducir los tiempos de inserción, evitando así reservas y borrado de memoria dinámica durante la ejecución.

También habría que remarcar que la aplicación de UNISTIM se apoya en el módulo de grabación de llamadas para la extracción de datos y resultados.

4.5 Conclusiones:

En este capítulo se ha mostrado como se ha desarrollado cada uno de los módulos de la aplicación. También se ha mostrado qué datos hemos obtenido según el paquete que nos llegaba y según en qué estado estuviésemos. Consiguiendo así extraer ficheros con información relevante de la conexión, tanto RTP como UNISTIM.

Por otro lado, se ha podido ver las altas dependencias que se tienen respecto al Gestor de Llamadas, ya que es el encargado de ordenar todo lo que tiene que ver con la señalización y el flujo. Que el terminal no pueda tomar sus propias decisiones hace que su parte sea más sencilla de monitorizar, mientras que la parte del Gestor hace que sea un poco más imprevisible y se tengan que tener en cuenta varias alternativas en sus decisiones a la hora de monitorizar el flujo UNISTIM.

También hemos comentado cómo este módulo de UNISTIM se introduce en un proyecto de disección y monitorización mucho mayor, VoIPCallMon. Ampliando así la funcionalidad del proyecto y mejorando las prestaciones del módulo UNISTIM.

Con esto damos por finalizada toda la parte de implementación y desarrollo. En el próximo capítulo se mostrarán las pruebas realizadas al conjunto desarrollado. Se crearán diversos escenarios para comprobar que la respuesta de nuestra aplicación corresponde con lo esperado.

5 Validación

5.1 Introducción:

Con la aplicación ya implementada y adaptada al proyecto VoIPCallMon es el momento de llevarla a prueba. Este paso es fundamental, ya que es dónde se pueden ver las limitaciones y prestaciones de la aplicación desarrollada.

Para comprobar el comportamiento se han intentado simular varios escenarios y ver así los resultados. Todo esto dentro, de las limitaciones existentes a la hora de crear los escenarios.

5.2 Montaje y configuración de escenarios:

Debido a que no se disponían de trazas de prueba existentes, se pasó a crear los escenarios de una manera artesanal. Se buscaba disponer de varias conexiones entre terminales dirigidos por un Gestor de Llamadas capaz de establecer llamadas mediante el protocolo UNISTIM. Todo esto dentro de las limitaciones existentes, ya que de manera artesanal se hace imposible simular una red de grandes dimensiones.

Para realizar esta tarea, se eligió como Gestor de Llamadas el software denominado Asterisk [8]. Es un programa de software libre que proporciona las funcionalidades correspondientes de una central telefónica (PBX) y que corre en Linux. Dentro de sus funcionalidades se encuentran características que anteriormente solo estaban disponibles en sistemas PBX propietarios. Como por ejemplo, buzón de voz, conferencias y distribución automática de llamadas. Pero lo más interesante de Asterisk es que reconoce muchos protocolos de VoIP, tanto estandarizados como propietarios, entre ellos UNISTIM. Por lo tanto, este gestor de llamadas proporciona todo lo que se necesitaba de un Gestor de Llamadas para la creación de nuestros escenarios.

Una vez teníamos el Gestor, solo nos faltaban los terminales. Los terminales elegidos fueron los *Softphones* 2050 de Avaya [9]. Hubo tres razones fundamentales para la elección de estos terminales. Primero, que eran capaces de soportar UNISTIM como protocolo de señalización. En segundo lugar, que pertenecían a la gama de productos de la empresa Avaya, la poseedora de los derechos del protocolo UNISTIM. Por último, que al ser *Softphone*, da la posibilidad de instalarlo en cualquier computadora que disponga del sistema operativo Windows.

Ya elegidos el Gestor y los Terminales, se tuvo que configurar el sistema para que se pudiera establecer conexión entre Terminales. Para ello, se tuvieron que modificar varios archivos de configuración de Asterisk, con el fin de que reconociese a los terminales dentro de su red de comunicaciones.

Los dos archivos que se tuvieron que modificar fueron unistim.conf y extensions.conf dentro de la configuración de Asterisk [10]. El archivo unistim.conf se utiliza para configurar el *Softphone* dentro del Gestor, para que reconozca al dispositivo. También dispone de ciertos parámetros de diseño configurables, como que poner en la pantalla mientras el *Softphone* no está en uso o los puertos RTP a utilizar, etc. Por ejemplo:

```
[i2050]
device=[MAC_Address]
line => 100
```

Mientras que el archivo extensions.conf se utiliza para crear una nueva extensión dentro del Gestor, para un determinado dispositivo. Para indicar así, cual va a ser el número del dispositivo. El ejemplo para el caso anterior sería:

```
exten=2002,1,Dial(USTM/100@i2050,20,tr)
```

Entonces, para llamar al *Softphone* con el nombre de i2050 habría que marcar el 2002. Añadiendo más dispositivos a los archivos de configuración se puede crear una red de terminales.

Sabido ya como montar una comunicación entre varios terminales con un Gestor de Llamadas, solo queda probar la aplicación.

5.3 Pruebas realizadas:

En este apartado se expondrán los diversos escenarios a los que hemos enfrentado a nuestra aplicación. Se ha ido incrementando el volumen de llamadas a medida que se comprobaba que todo funcionaba según lo esperado. Además se han hecho pruebas con llamadas que no se realizaron correctamente, para disponer así de todas las posibilidades.

El número máximo de terminales al que se llegó a probar es de seis conectados simultáneamente, siendo un número razonable para comprobar las conexiones entre varios terminales.

Como último paso antes de entrar en los escenarios, habría que realizar una aclaración sobre el camino que hace el RTP. Hemos visto en el estado del arte, que en la mayoría de casos en los que se utiliza VoIP, el flujo RTP viaja de extremo a extremo. En el caso de Asterisk el tráfico viaja a través del propio Gestor. Durante el estudio previo también se pudo comprobar que existe la posibilidad de que viaje de extremo a extremo, pero no es el caso de Asterisk.

5.3.1 Escenario 1: Conexión 1 a 1.

En este primer escenario, simplemente se prueba el funcionamiento de la aplicación mientras se establece una conexión. Una conexión que se realiza correctamente.

El escenario sería el siguiente:

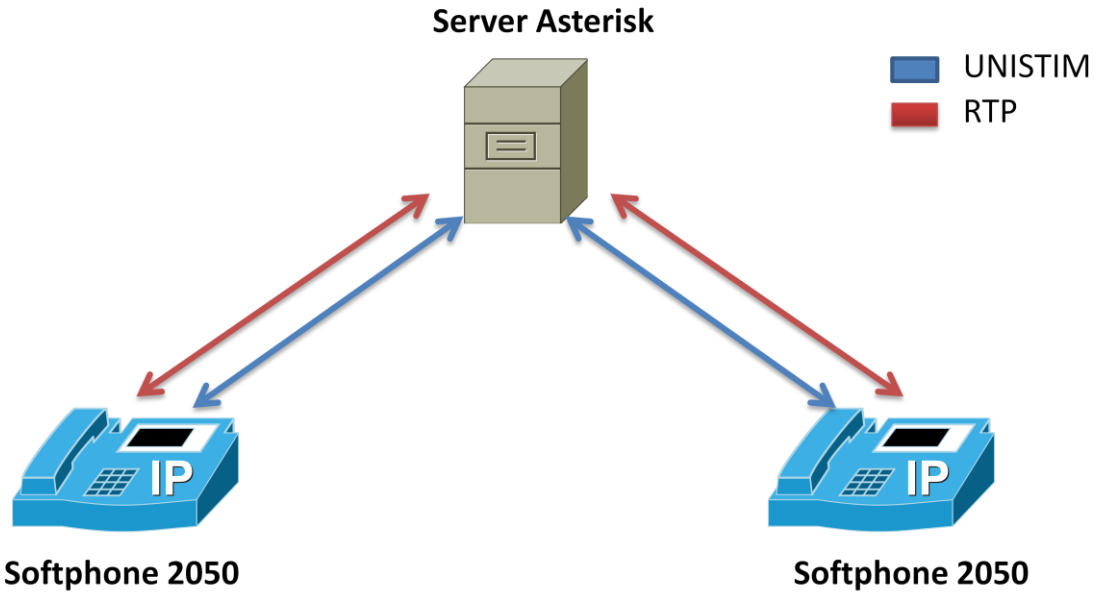

Figura X: Escenario 1- Conexión 1 a 1

Los resultados obtenidos son:

- **Fichero de extracción de información UNISTIM**: Extrae la información monitorizada de los dos flujos UNISTIM. Los dos flujos de señalización que mantiene el Gestor de Llamadas con los dos Terminales.

- **Fichero de extracción de información RTP:** También nos muestra la información de dos flujos, en este caso, la de los flujos de audio RTP. Es decir, de ambos extremos al Gestor, tal y como se ve en la figura.

- **Capturas del tipo pcap:** Obtenemos dos ficheros del tipo pcap, en el que se encuentran todos los paquetes del tráfico UNISTIM. Uno por cada conexión con el Gestor.

- **Grabaciones de audio**: Obtenemos dos carpetas, una por cada extremo. Dentro de ellas se encuentra el audio dividido en dos partes, el audio de ida y el audio de vuelta. Ambos ficheros se pueden combinar para poder escuchar la llamada completa.

5.3.2 Escenario 2: Conexión fallida + contestador Asterisk.

Una vez probado en una conexión que se realiza correctamente, es necesario probarlo en una conexión que no se llega a realizar, para comprobar que el comportamiento no es el mismo que si se hubiese realizado correctamente.

Este es el caso de llamadas en las que se marca un número que se encuentra ocupado, o un número que no tiene registrado el Gestor. Hay que recalcar, que en estos casos Asterisk tiene configurado un contestador para informar al usuario.

El escenario es el siguiente:

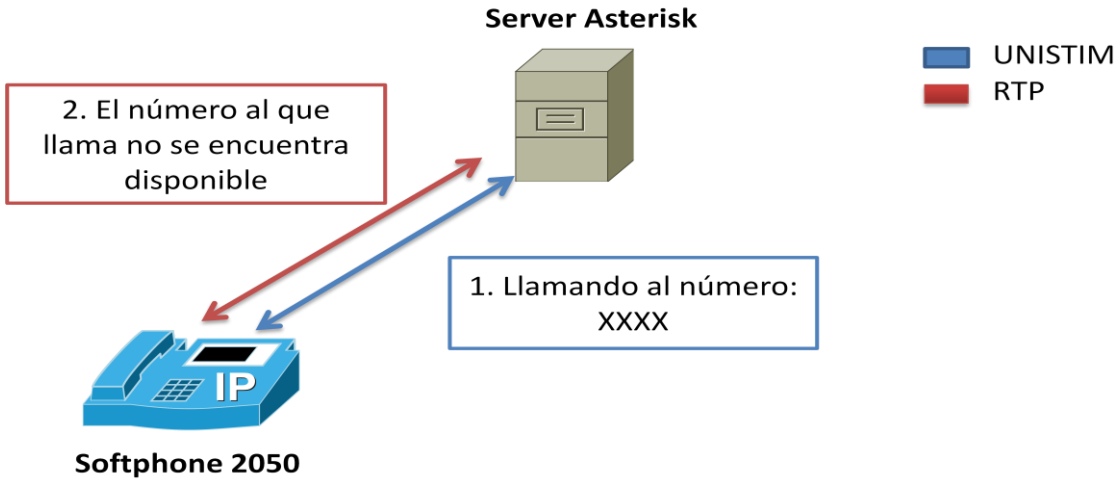

Figura XI: Escenario 2: conexión fallida + contestador Asterisk

Los resultados obtenidos son:

- **Fichero de extracción de información UNISTIM**: Extrae la información monitorizada de un solo flujo UNISTIM. El que se produce entre Terminal y Gestor de Llamadas. Es decir, lo esperado, ya que no es posible la conexión.

- **Fichero de extracción de información RTP:** Ocurre lo mismo que en UNISTIM. Solo se extrae información del flujo entre el Terminal y el Gestor. Este flujo se establece debido a que Asterisk tiene configurado una voz, que informa de que la llamada no se pudo realizar al usuario del terminal. De manera diferente a lo que ocurría en el escenario 1, aquí solo se establece un flujo RTP.

- **Capturas del tipo pcap:** En este caso obtenemos un único fichero pcap, ya que solo tenemos una conexión. El terminal que realizó la llamada con el Gestor. En esta captura, se puede comprobar que una vez el Gestor informa de la situación al usuario mediante el flujo de audio, se cierra el flujo mediante un mensaje del tipo *Close Audio Stream*.

- **Grabaciones de audio**: Solo obtenemos una carpeta. Dentro de ella se encuentra el audio dividido en el audio de ida y el audio de vuelta.

5.3.3 Escenario 3: Conexión 2 a 2 + error en llamada.

En este escenario se intentó realizar dos conexiones entre terminales de manera simultánea. Pero se descubrió que el Gestor tuvo problemas para establecer el RTP de la llamada, por lo que no pudo fluir la voz de extremo a extremo. Fue debido a un error del propio Asterisk ajeno a nuestra aplicación, ya que no pudo establecer la conexión debido a algún error en la configuración o en el enlace por causas desconocidas. Aún así, se decidió guardar la prueba para comprobar el comportamiento de la aplicación ante un fallo así.

El escenario que quedaría es el siguiente:

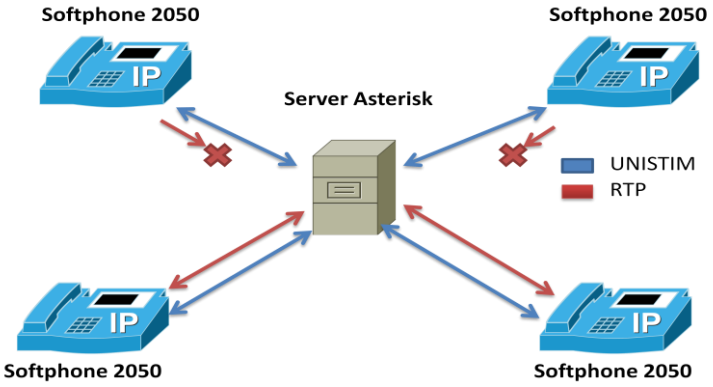

Figura XII: Escenario 3- Conexión 2 a 2 + error en llamada

Los resultados obtenidos son:

- **Fichero de extracción de información UNISTIM**: Se contemplan los cuatro flujos UNISTIM correspondientes. Es decir, la comunicación de los terminales con el Gestor.

- **Fichero de extracción de información RTP:** En este fichero, en lugar de cuatro flujos RTP, que se hubiesen producido si ambas conexiones se hubiesen completado, tenemos dos flujos únicamente, tal y como se ve en la figura XII. Es decir, encontramos una anomalía, también, tras el paso por nuestra aplicación.

- **Capturas del tipo pcap:** En cuanto a las capturas de tráfico UNISTIM y almacenaje en ficheros pcap, se realizan de manera adecuada. Esto son cuatro ficheros con la correspondiente señalización.

- **Grabaciones de audio**: Mientras que en las grabaciones, al no tener flujo RTP solo disponemos de las carpetas correspondientes a la que funcionó de manera correcta. Es decir, también encontramos la anomalía.

5.3.4 Escenario 4: Conexión 3 a 3.

Como último escenario, se hará la probatura de tener tres conexiones de manera simultánea bajo la monitorización de la aplicación. En este caso se probó con conexiones que se realizaron correctamente. Con el fin de aumentar la carga de trabajo del módulo UNISTIM y comprobar su funcionamiento.

El escenario creado sería como este:

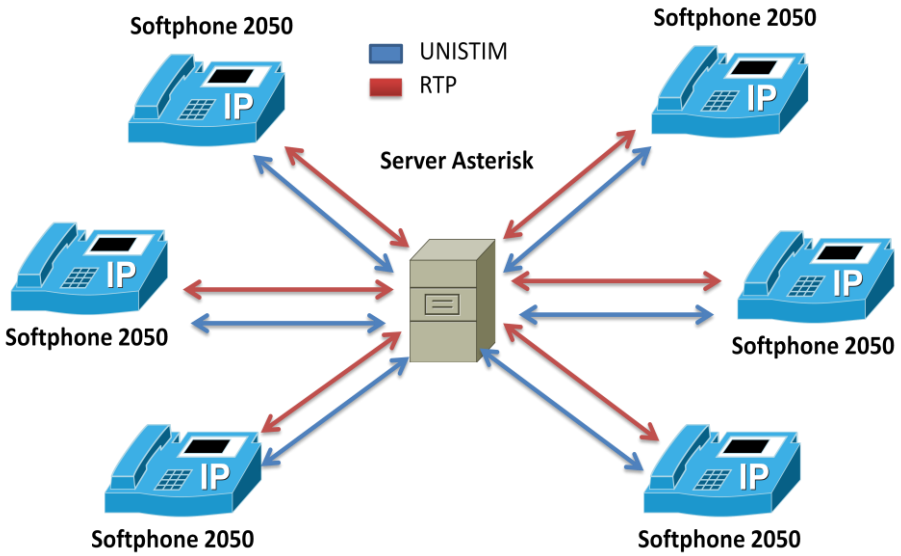

Figura XIII: Escenario 4- Conexión 3 a 3

Los resultados obtenidos son:

- **Fichero de extracción de información UNISTIM**: En el fichero de texto plano que extrae la información de los enlaces del protocolo de señalización UNISTIM, comprobamos que se han conseguido monitorizar los seis flujos. Tal y como se ve en la figura.

- **Fichero de extracción de información RTP:** Al igual que en el fichero de UNISTIM, se consiguen monitorizar y detectar todos los flujos de voz que mantienen todos los terminales con el Gestor.

- **Capturas del tipo pcap:** También se puede comprobar que se almacena el tráfico UNISTIM en ficheros del tipo pcap de manera correcta. Es decir, los seis ficheros esperados, uno por cada flujo.

- **Grabaciones de audio**: Por último, nos cercioramos de que hay seis carpetas con el audio correspondiente. En cada carpeta se encuentra el audio que ha enviado y recibido cada terminal.

5.4 Conclusiones:

Visto el comportamiento de la aplicación ante todos esos escenarios, podemos decir que se comporta según lo esperado. Obteniendo toda la información requerida del protocolo de señalización y del flujo de audio.

Por otro lado, hemos visto que si se produce un comportamiento extraño en alguna de las conexiones, observando los resultados que nos brinda la aplicación podemos detectarlos. Algo que resultaría muy útil para futuras funcionalidades del proyecto.

Por último, y para finalizar el documento, se mostrarán las conclusiones extraídas del trabajo realizado. Además se enumerarán algunas ideas para llevar a cabo en el futuro para perfeccionar la aplicación y brindarla de más funcionalidad.

6 Conclusiones y trabajo futuro

6.1 Introducción:

En el último capítulo de este Trabajo Fin de Grado se analizará la aplicación desarrollada y se extraerán las conclusiones finales del documento. Además, se mencionaran los puntos que aún tienen un margen de mejora, y que podrían suponer un punto de partida para otros trabajos, con el fin de ampliar la funcionalidad de la aplicación y del proyecto. Esto se tratará en el apartado de Trabajo Futuro.

6.2 Conclusiones:

El objetivo de este Trabajo Fin de Grado era desarrollar un sistema de medida de recogida de datos y monitorización de tráfico VoIP del protocolo UNISTIM. El resultado ha sido una aplicación capaz de diseccionar y monitorizar conexiones que utilicen como protocolo de señalización UNISTIM. Además es una aplicación adaptada a un proyecto de monitorización mucho mayor, que junto con el modulo creado, se ha conseguido una funcionalidad mucho mayor en su conjunto.

La aplicación se dividió en tres módulos: captura del tráfico, análisis UNISTIM y extracción de resultados de la monitorización. Con esta división se ha conseguido estructurar la solución con el fin de que fuese más sencillo de comprender. También ayudó al desarrollo del código, llevar a prueba a la aplicación y a integrarlo en VoIPCallMon.

Una vez fijados los requisitos, se ha conseguido implementar una aplicación capaz de diseccionar cualquier paquete del protocolo de VoIP UNISTIM. Siguiendo la lógica desarrollada, es capaz de tomar decisiones a la hora de la obtención de datos. Sabiendo, eso sí, en qué momento de la llamada se encuentra y que es lo que esperamos en cada momento. Además, puede identificar cuando se inicia el flujo RTP para capturarlo.

Por otro lado, se ha conseguido adaptar las prestaciones de nuestra aplicación al proyecto VoIPCallMon. Con el fin de mejorar la funcionalidad en el proyecto y en la propia aplicación. El módulo UNISTIM consigue adaptarse a unas estructuras eficientes en el uso de memoria, con el fin de ganar velocidad de procesado, y VoIPCallMon aumenta su funcionalidad, pudiendo monitorizar un protocolo que antes no podía. Creándose así, una simbiosis muy beneficiosa para ambas partes y mejorando sus prestaciones.

Posteriormente, una vez la aplicación estuvo operativa se puedo probar bajo varios escenarios. El problema que se tenía en este punto, era que al tener que montar nosotros mismos el escenario, no había posibilidad de crear un entorno con muchas conexiones. Por lo tanto, teniendo en cuenta las limitaciones que se tuvieron a la hora de realizar las pruebas, ya que no se tenía forma de probar la aplicación a gran escala, se comprobó

que todo funcionaba como se esperaba. Se ha comprobado que el módulo funciona para varias llamadas de manera simultánea y que es capaz de detectar cuando se produce tráfico RTP, en el caso de que se haya establecido la señalización correctamente. También se ha comprobado que en el caso de que no se haya establecido flujo RTP, la aplicación no almacena nada de flujo de audio, pero sí guarda la información del flujo UNISTIM. En definitiva, se obtiene toda la información requerida para la monitorización y disección de un enlace que utilice UNISTIM.

Finalmente, se ha realizado una aplicación con unos buenos resultados, obteniendo lo esperado, y se ha extraído una experiencia muy enriquecedora de la realización de este Trabajo Fin de Grado. Se ha aprendido sobre el mundo de la voz sobre IP y sobre estructuras de programación, a cómo usarlas en beneficio del proyecto. Pero aún así, ningún trabajo es totalmente perfecto y se pueden realizar mejoras que ayuden a aumentar la funcionalidad del proyecto y conseguir que sea más completo. El conjunto de mejoras que se podrían llevar a cabo las propongo en el apartado siguiente.

6.3 Trabajo futuro:

Para ampliar la funcionalidad de este Trabajo Fin de Grado o sus puntos débiles se deja una posible lista de trabajos a realizar en el futuro. Con el fin de mejorar el conjunto de la aplicación. Entre ellos se encuentran:

- Realizar pruebas de la aplicación bajo una alta demanda de conexiones que funcionen bajo el protocolo de UNISTIM. Una demanda parecida a la que podría producirse en una gran empresa entre sus teléfonos corporativos.

- Crear un interfaz capaz de recibir datos de las conexiones, con el fin de crear estadísticas y medidas útiles para usuarios, administradores de la red o cualquier otro agente. Con una gestión de usuarios, consultas de datos de llamadas antiguas, capacidad de descargar una grabación de la llamada.

- Crear un sistema que, mediante la utilización de los datos extraídos con el módulo de UNISTIM, sea capaz de identificar de manera instantánea que se ha producido un problema en una conexión determinada.

- Repartir varias sondas de monitorización a lo largo de un enlace para obtener medidas desde varios puntos de la conexión, para poder realizar comparaciones en la señalización del protocolo de VoIP y el flujo de audio. Pudiendo así detectar problemas en determinados puntos.

- Estudio e implementación de otros protocolos de VoIP, no disponibles en la actualidad, dentro de VoIPCallMon. Ya sean protocolos estandarizados como propietarios, buscando una funcionalidad mucho mayor del proyecto.

BIBLIOGRAFÍA

[1] Choi Y, Silvester J, Kim H-C. Analyzing and modeling workload characteristics in a multiservice IP network. IEEE Internet Computing 2011.

[2] Birke R, Mellia M, Petracca M, Rossi D. Experiences of VoIP traffic monitoring in a commercial ISP. International Journal of Network Management 2010.

[3] VoIP – Telefonía IP. Disponible en http://elastixtech.com

[4] Forefront TMG is SIP – aware. Disponible en http://blogs.technet.com/

[5] Karapantazis S, Pavlidou F. VoIP: a comprehensive survey on a promising technology. Computer Networks 2009;

[6] William A, Robert S, Kris W, Larry J, Kenneth M, Peter P, Robert M, Kenneth J, Dariusz O, Paul P, Robert J. Telephony and data network services at a telephone. Patent US 7068641 B1.

[7] José G, Pedro M, Javier R, David M, Víctor M, Jorge E, Javier A. Lowcost and high-performance: VoIP monitoring and full-data retention at multi-Gb/s rates using commodity hardware. International Journal of Network Management 2014.

[8] Jim V, Leif M, Jared S. Asterisk: The future of Telephony. Second Edition.

[9] Avaya. Guía del usuario del teléfono IP Softphone 2050 de Avaya. Año 2010.

[10] chan_unistim. Dispoible en http://www.voip-info.org/